Removing Bidirectionality from Nondeterministic Finite Automata

Christos Kapoutsis

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
cak@mit.edu

Abstract. We prove that every *two-way* nondeterministic finite automaton with n states has an equivalent *one-way* nondeterministic finite automaton with at most $\binom{2n}{n+1}$ states. We also show this bound is exact.

1 Introduction

Converting an arbitrary *one-way nondeterministic finite automaton* (1NFA) to an equivalent *one-way deterministic finite automaton* (1DFA) has long been the archetypal problem of descriptional complexity. Rabin and Scott [1][2] proved that starting with an n-state 1NFA one can always construct an equivalent 1DFA with at most $2^n - 1$ states;[1] later observations [3, 4][5–8] established the tightness of this upper bound, in the strong sense that, for all n, some n-state 1NFA has no equivalent 1DFA with fewer than $2^n - 1$ states. So, we often say that the *trade-off* from 1NFAs to 1DFAs is *exactly* $2^n - 1$. (Fig. 1*a*.)

The fact that this problem initiated the discussion on issues of descriptional complexity is only one aspect of its significance. A more interesting aspect is that its solution fully uncovered and elegantly described the relationship between the computations of the two types of machines. This is supported not only by the fact that the demonstrated upper and lower bounds match *exactly* (as opposed to merely asymptotically), but also —and more crucially— by the central role that a well-understood set-theoretic object plays in the associated proof: what the theorem really tells us is that every 1NFA N can be simulated by a 1DFA that has one distinct state for each *non-empty subset of states of* N which (as an instantaneous description of N) is both realizable and irreplaceable. From this, the demonstrated trade-off is then only a counting argument away, plus a clever search for 1NFAs that indeed manage to keep all of their instantaneous descriptions realizable and irreplaceable.

In the present study we offer a similar analysis for the conversion of an arbitrary *two-way nondeterministic finite automaton* (2NFA) to a one-way equivalent: we prove that the trade-off from 2NFAs to 1NFAs is exactly $\binom{2n}{n+1}$. As above, we first identify the correct set-theoretic object that 'lives' in the relation between 2NFA and 1NFA computations, and then easily extract the trade-off.

[1] In this article, all finite automata are allowed to be *incomplete*: their transition functions may be partial, and thus computations may hang inside the input.

Two-way finite automata were introduced in the late 50's [9, 2, 10] and shown equivalent to 1DFAs. Originally, their definition included neither endmarkers nor nondeterminism, so they were just *single-pass two-way deterministic finite automata* (ZDFAs). However, they soon grew into full-fledged *two-way deterministic finite automata* (2DFAs) and nondeterministic versions (ZNFAs and 2NFAs), which all remained equivalent to 1DFAs. Since then, the cost of the 2NFA-to-1NFA conversion has been addressed sporadically.

Shepherdson's proof [10] implied every n-state 2NFA can be converted into a 1NFA with at most $n2^{n^2}$ states. A cheaper simluation via crossing sequences [11, Sect. 2.6] has also been known, with only $O(2^{2n \lg n})$ states. However, a straightforward elaboration on [10] shows the cost can be brought down to even $n(n+1)^n$. Which still wastes exponentially many states, as Birget [12] showed $8^n + 2$ are always enough, via an argument based on length-preserving homomorphisms. Here, we establish the still exponentially smaller upper bound of $\binom{2n}{n+1}$.

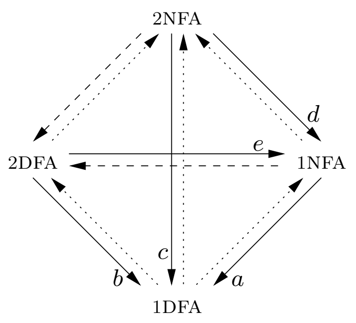
On the other hand, exponential lower bounds have also been known, even when the starting automaton is deterministic [7, 12] and single-pass [6, 13]. For example, Damanik [13] gives a language that costs $\leq 4n + 2$ on ZDFAs, but $\geq 2^n$ on 1NFAs. Here, we give a lower bound that matches $\binom{2n}{n+1}$, even when we start from a ZDFA. Hence, *the ability of a* 2NFA *to move its head bidirectionally* strictly inside the input *can alone cause all the hardness a simulating* 1NFA *must subdue.*

The conversions from 1NFAs to 1DFAs and from 2NFAs to 1NFAs are only two of a dozen different conversions among the four basic automata models (1DFAs, 1NFAs, 2DFAs, and 2NFAs) for the regular languages. Each of the 12 arrows in Fig. 1 represents one of these conversions and defines the associated problem of finding the exact trade-off. Of the 12 problems, some are little harder than clever exercises, but others are decades-old open questions on the power of nondeterminism —a surprising range in difficulty. We present a quick review.

The arguments of [2–4] and this study establish that $a = 2^n - 1$ and $d = e = \binom{2n}{n+1}$, while an argument of [14] shows the trade-off for every conversion from a weaker to a stronger model (dotted arrows) is exactly $f = n$. From 2DFAs and 2NFAs to 1DFAs, it has been known that the trade-offs are exponential [10, 15, 4, 5, 7, 16, 12], although the exact values remained elusive; refining [12] and following the rational of this study, we can show them to be as in Fig. 1 (the proofs to appear in the full version of this article). This leaves only the questions for the conversions from 1NFAs and 2NFAs to 2DFAs (dashed arrows), which remain wide open: more than 30 years after they were first asked [6], not only are the exact trade-offs unkown, but we cannot even confirm the conjecture that they are exponential (see [17] for a discussion).

Finally, we should note that Fig. 1 shows only four of the numerous automata that solve exactly the regular languages. Bidirectionality, nondeterminism, alternation, randomness, pebbles, and other enhancements, alone or combined, limited or unrestricted, give rise to a long list of variants and to the corresponding descriptional-complexity questions. See [18] for a comprehensive overview.

The next section defines the basic concepts. Section 3 establishes the upper bound, while Sect. 4 proves that it is exact. We conclude in Sect. 5.



$$a = 2^n - 1$$
$$b = n\big(n^n - (n-1)^n\big)$$
$$c = \textstyle\sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \binom{n}{i}\binom{n}{j}\left(2^i - 1\right)^j$$
$$d = e = \binom{2n}{n+1}$$
$$f = n$$

Fig. 1. Trade-off summary ($f = n$ on all dotted arrows; dashed arrows are open).

2 Preliminaries

We write $[n]$ for the set $\{1, 2, \ldots, n\}$. The special objects $\mathtt{l}$, $\mathtt{r}$, $\perp$ are used for building the *disjoint union* of two sets and the *augmentation* of a set

$$A \uplus B = (A \times \{\mathtt{l}\}) \cup (B \times \{\mathtt{r}\}) \qquad \text{and} \qquad A_\perp = A \cup \{\perp\}.$$

When A and B are disjoint, $A \cup B$ is also written as $A + B$. The size of A is denoted by $|A|$, while $(A \to B)$ denotes the set of functions from A to B.

For Σ an alphabet, we use Σ^* for the set of all finite strings over Σ and Σ_e for $\Sigma + \{\vdash, \dashv\}$, where $\vdash$ and $\dashv$ are special endmarking symbols. If w is a string, $|w|$ is its length and w_i is its i-th symbol, for $i = 1, \ldots, |w|$. The 'i-th *boundary* of w' is the boundary between w_i and w_{i+1}, if $i = 1, \ldots, |w| - 1$; or the leftmost (rightmost) boundary of w, if $i = 0$ ($i = |w|$). (Fig. 2a.) We also write w_e for the extension $\vdash w \dashv$ of w and $w_{\mathrm{e},i}$ for $(w_\mathrm{e})_i$. The empty string is denoted by ϵ.

We present 2NFAs and 1NFAs as variations of the more natural model of a 2DFA. The next paragraph defines this model and some basic relevant concepts.

Two-Way Deterministic Finite Automata. We assume the reader is familiar with the intuitive notion of a 2DFA. Formally, this is a triple $M = (s, \delta, f)$, where δ is the *transition function*, partially mapping $Q \times \Sigma_\mathrm{e}$ to $Q \times \{\mathtt{l}, \mathtt{r}\}$, for a set Q of *states* and an alphabet Σ, while s, f are the *start* and *final* states. Here, we insist the automaton *accepts* only if it moves past the right endmarker $\dashv$ into f, this being the only case in which an endmarker may be violated.[2]

Computations. Although M is typically started at s and on $\vdash$, many other possibilities exist: for any w, i, q, the *computation of M when started at state q on the i-th symbol of string w* is the unique sequence

$$\text{COMP}_{M,q,i}(w) = \big((q_t, i_t)\big)_{0 \le t \le m}$$

[2] So, on $\vdash$, δ moves right or hangs. On $\dashv$, it moves left, hangs, or moves right into f.

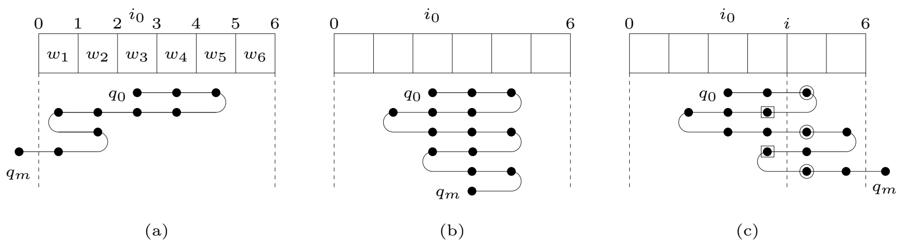


Fig. 2. (a) Cells and boundaries on a 6-long w; a computation that hits left. (b) One that hangs. (c) One that hits right, and its i-th frontier: R^c_i in circles and L^c_i in boxes.

where $(q_0, i_0) = (q, i)$, $0 \leq m \leq \infty$, every pair is derived from its predecessor via δ and w, every pair is within w $(1 \leq i_t \leq |w|)$ except possibly for the last one, and the last pair is within w iff δ is undefined on the corresponding state and symbol. We say (q_t, i_t) is the *t-th point* and m the *length* of this computation. If $m = \infty$, the computation *loops*. Otherwise, it *hits left into q_m*, if $i_m = 0$; or *hangs*, if $1 \leq i_m \leq |w|$; or *hits right into q_m*, if $i_m = |w| + 1$ (Fig. 2). When $i = 1$ (respectively, $i = |w|$) we get the *left (right) computation of M from q on w*:[3]

$$\text{LCOMP}_{M,q}(w) ::= \text{COMP}_{M,q,1}(w) \quad \text{and} \quad \text{RCOMP}_{M,q}(w) ::= \text{COMP}_{M,q,|w|}(w).$$

The *computation of M on w* refers to the typical $\text{COMP}_M(w) ::= \text{LCOMP}_{M,s}(w_e)$, so that M accepts $w \in \Sigma^*$ iff $\text{COMP}_M(w)$ hits right (into f). Note that, since M can violate an endmarker only when it moves past $\dashv$ into f, a computation of M on an endmarked string can only loop, or hang, or hit right into f.

Frontiers. Fix a computation $c = ((q_t, i_t))_{0 \leq t \leq m}$ and consider the i-th boundary of the input (Fig. 2c). This is crossed ≥ 0 times. Collect into a set R^c_i (respectively, L^c_i) all states that result from a left-to-right (right-to-left) crossing,

$$R^c_i = \{q_{t+1} \mid 0 \leq t < m \ \& \ i_t = i \ \& \ i_{t+1} = i + 1\},$$
$$L^c_i = \{q_{t+1} \mid 0 \leq t < m \ \& \ i_t = i + 1 \ \& \ i_{t+1} = i\},$$

also requiring that $R^c_{i_0-1}$ contains q_0.[4] The pair (L^c_i, R^c_i) partially describes the behavior of c over the i-th boundary and we call it the *i-th frontier of c*. Note that the description is indeed partial, as the pair contains no information on the order in which c exhibits the states around the boundary or on number of times each state is exhibited. For a full description one would need instead the *i-th crossing sequence of c* [11]. However, for our purposes, the extra information provided by the complete description is redundant.

[3] Note that, when w is the empty string, the left computation of M from q on w is just $\text{LCOMP}_{M,q}(\epsilon) = \big((q, 1)\big)$ and therefore hits *right* into q, whereas the corresponding right computation $\text{RCOMP}_{M,q}(\epsilon) = \big((q, 0)\big)$ hits *left* into q.

[4] This reflects the convention that the starting state of any computation is considered to be the result of an 'invisible' left-to-right step.

Variations. If in the definition of M above more than one next moves are allowed at each step, we say the automaton is *nondeterministic* (a 2NFA). This formally means that δ *totally* maps $Q \times \Sigma_e$ to the *powerset* of $Q \times \{\mathtt{l}, \mathtt{r}\}$ and implies that $C = \text{COMP}_{M,q,i}(w)$ is now a *set* of computations. If then

$$P = \{p \mid \text{some } c \in C \text{ hits right into } p\},$$

we say C *hits right into* P; and w is accepted iff $\text{COMP}_M(w)$ hits right into $\{f\}$.

If the head of M never moves to the left, we say M is *one-way* (a 1NFA).[5] If no computation of M 'continues after arriving at an endmarker', we say M is *single-pass* (a ZNFA; or a ZDFA, if M is deterministic).

3 The Upper Bound

Fix an n-state 2NFA $N = (s, \delta, f)$ over alphabet Σ and state set Q. In this section we build an equivalent $\binom{2n}{n+1}$-state 1NFA N' via an optimal construction.

Frontiers. Assume momentarily that N is deterministic and $c = \text{COMP}_N(w)$ is accepting, for some l-long input w. Consider the i-th frontier (L_i^c, R_i^c) of c, for some $i \neq 0, l+2$. The number of states in R_i^c equals the number of times c left-to-right crosses the i-th boundary: each crossing contributes a state into R_i^c and no two crossings contribute the same state, or else c would be looping. Similarly, $|L_i^c|$ equals the number of times c right-to-left crosses the i-th boundary. Now, since c accepts, it goes from $\vdash$ all the way past $\dashv$, forcing the rightward crossings on every boundary to be exactly 1 more than the leftward crossings. Hence,

$$|L_i^c| + 1 = |R_i^c|,$$

which remains true even on the leftmost boundary ($i = 0$, under our convention from Footn. 4) and also on the rightmost one ($i = l + 2$). So, the equality holds over every boundary and motivates the following definition.

Definition 1. *A* frontier of N *is any* (L, R) *with* $L, R \subseteq Q$ *and* $|L| + 1 = |R|$.

So, *if the computation of a* 2DFA *on a particular input is accepting, then all frontiers of the computation are frontiers of this* 2DFA.

For our nondeterministic N, though, the argument breaks, as a state repetition under a cell may not imply looping. However, it does imply a cycle. So, let us call a computation *minimal* if it contains no cycles (i.e., if every two of its points are distinct) and repeat the previous argument to establish the following.

Lemma 1. *All frontiers of an* accepting minimal *computation of N on some input are frontiers of N.*

[5] Note that our 1NFAs work on *endmarked* inputs, a deviation from typical definitions.

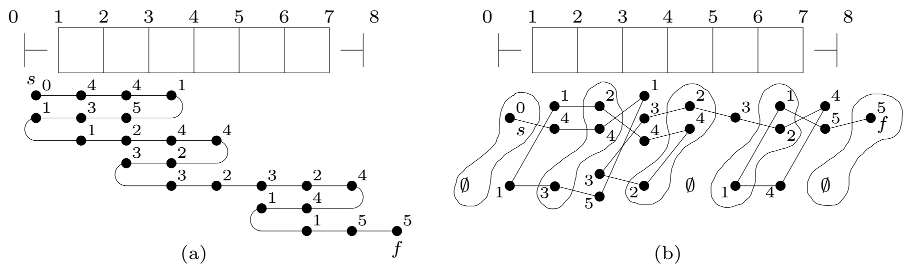


Fig. 3. (a) An accepting minimal $c \in \textsc{comp}_N(w)$, for $|w| = 6$, state set $\{0, 1, \dots, 5\}$, $s = 0$, $f = 5$. (b) The same c arranged in frontiers; the even-indexed ones are circled.

Compatibilities among Frontiers. Suppose c is an accepting minimal computation of N on an l-long w and let $F^c_i = (L^c_i, R^c_i)$ be its *i-th frontier*, for each $i = 0, 1, \dots, l+2$ (Fig. 3). Note that the first and last frontiers are always

$$F^c_0 = (\emptyset, \{s\}) \qquad \text{and} \qquad F^c_{l+2} = (\emptyset, \{f\}),$$

as c starts at s, ends in f, and never right-to-left crosses an outer boundary.

Also note that, for $(L, R) = (L^c_i, R^c_i)$ and $(L', R') = (L^c_{i+1}, R^c_{i+1})$ two successive frontiers (Fig. 4a), it should always be that $R \cap L' = \emptyset$: otherwise, c would be repeating a state under w_i and would not be minimal. Hence, $R + L'$ contains as many states as many (occurences of) states there are in L and R' together:

$$|R + L'| = |R| + |L'| = \big(|L| + 1\big) + \big(|R'| - 1\big) = |L| + |R'| = |L \uplus R'|.$$

So, bijections can be found from $R + L'$ to $L \uplus R'$. Among them, a very natural one (Fig. 4b): for each $q \in R + L'$ find the unique step in c that produces q under w_i (this is either a rightward crossing of boundary i or a leftward crossing of boundary $i + 1$; the minimality of c guarantees uniqueness); the next step left-to-right crosses boundary $i + 1$ into some state $p \in R'$ or right-to-left crosses boundary i into some $p \in L$; depending on the case, map q to $(p, \mathtt{r})$ or $(p, \mathtt{l})$ respectively. If $\rho : R + L' \to L \uplus R'$ is this mapping, it is easy to verify that it is injective (because c is minimal) and therefore bijective, as promised. In addition, ρ clearly respects the transition function: $\rho(q) \in \delta(q, w_i)$, for all $q \in R + L'$.[6]

Overall, we see that the sequence of frontiers exhibited by an accepting minimal $c \in \textsc{comp}_N(w)$ obeys some restrictions. We now formally summarize them.

Definition 2. *Let (L, R), (L', R') be frontiers of N and $a \in \Sigma_{\mathrm{e}}$. We say that (L, R) is* a-compatible *to (L', R') iff $R \cap L' = \emptyset$ and some bijection $\rho : R + L' \to L \uplus R'$ respects the transition function on a: for all $q \in R + L'$: $\rho(q) \in \delta(q, a)$.*

Definition 3. *Suppose $w \in \Sigma^*$ is l-long and $F_0, F_1, \dots, F_{l+2}$ is a sequence of frontiers of N. We say the sequence* fits w *iff*

[6] Throughout this argument, w_i really refers to $w_{\mathrm{e},i+1}$. This is w_i only when $i \neq 0, l+1$.

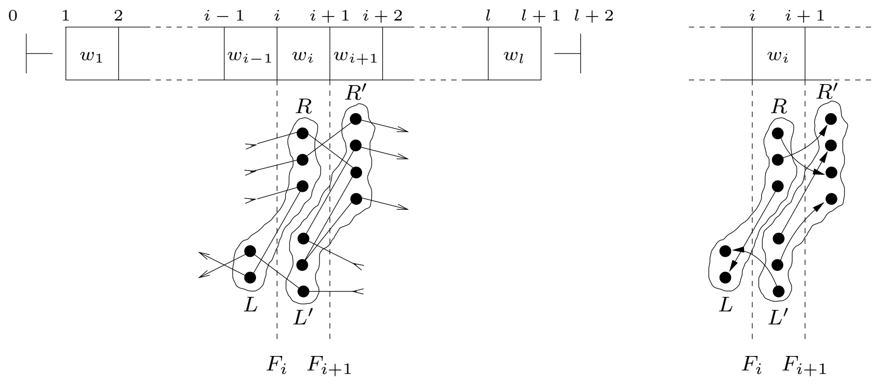


Fig. 4. (a) Two successive frontiers, on the left. (b) The associated ρ, on the right.

1. *$F_0 = (\emptyset, \{s\})$,*
2. *for all $i = 0, 1, \ldots, l+1$: F_i is $w_{\mathrm{e},i+1}$-compatible to F_{i+1},*
3. *$F_{l+2} = (\emptyset, \{f\})$.*

Lemma 2. *If* $\textsc{comp}_N(w)$ *contains an accepting computation, then some sequence of frontiers of N fits w.*

Proof. Every accepting computation gives rise to a *minimal* accepting one. $\square$

The Main Observation. The converse of Lemma 2 is also true: *if a sequence of frontiers of N fits w, then* $\textsc{comp}_N(w)$ *contains an accepting computation.*

To prove this, fix an l-long w and assume some sequence of frontiers of N

$$F_0 = (L_0, R_0), \quad F_1 = (L_1, R_1), \quad \ldots, \quad F_{l+2} = (L_{l+2}, R_{l+2})$$

fits w. We show the stronger claim that, for every i, the states of R_i can be produced by $|R_i|$ right-hitting computations on $\vdash w_1 \cdots w_{i-1}$: one starting at s and on $\vdash$, each of the others starting at some $q \in L_i$ and on w_{i-1}.

Claim. For all $i = 0, 1, \ldots, l+2$, some bijection $\pi_i : (L_i)_\perp \to R_i$ is such that
1. some $c \in \textsc{lcomp}_{N,s}(\vdash w_1 \cdots w_{i-1})$ hits right into $\pi_i(\perp)$, and
2. for all $q \in L_i$, some $c \in \textsc{rcomp}_{N,q}(\vdash w_1 \cdots w_{i-1})$ hits right into $\pi_i(q)$.

(Here, we take w_0 and w_{l+1} to mean the endmarkers $\vdash$ and $\dashv$, respectively.) Note that our main observation follows from this claim for $i = l+2$.

To prove the claim, we use induction on i. The base case $i = 0$ is trivial. For the inductive step (Fig. 5a), assume $i < l+2$, let $(L, R) = (L_i, R_i)$, $(L', R') = (L_{i+1}, R_{i+1})$, $a = w_{\mathrm{e},i+1}$, and consider the bijections guaranteed by the inductive hypothesis, $\pi = \pi_i : L_\perp \to R$, and the fact that (L, R) is a-compatible to (L', R'), $\rho : R + L' \to L \uplus R'$. Based on π, ρ and a third function σ, we build a bijection $\pi' = \pi_{i+1} : (L')_\perp \to R'$ that satisfies (1), (2) of the claim. First, we introduce σ.

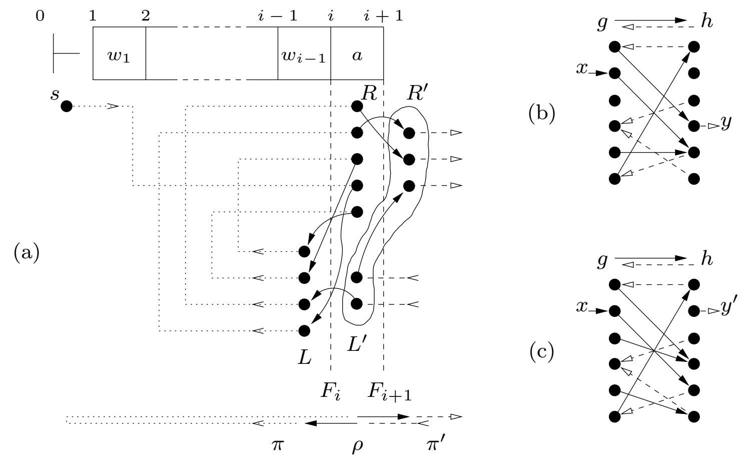


Fig. 5. (a) An example for the inductive step for the main observation. E.g., note that σ maps the 3rd and 5th (from top) states of R to $\perp$, while the 4th state is mapped to the 1st state of R'. (b) A nice input, that has a path. (c) A nice input with no path.

Definition of σ: pick some $q \in R$ and take a trip around under $\vdash w_1 w_2 \cdots w_{i-1} a$

$$\underset{r_0}{q},\ \underset{r_1}{\rho(q)},\ \underset{r_2}{\pi\rho(q)},\ \underset{r_3}{\rho\pi\rho(q)},\ \underset{r_4}{\pi\rho\pi\rho(q)},\ \ldots \tag{1}$$

by alternately following ρ and π, until the first time 'ρ fails to return a state in L'.[7] Let $r_0, r_1, r_2, \ldots$ be the states that we visit. We distinguish two cases.

Case 1: ρ does eventually fail to return a state in L. Then the trip is finite: $r_0, r_1, \ldots, r_k$, for an even $k \geq 0$ and an $r_k \in R$ that is ρ-mapped to some $q' \in R'$.

Case 2: ρ always returns a state in L. Then the trip is infinite and (since all even-indexed r_i and all odd-indexed r_i are respectively inside the finite sets R and L) there exist repetitions of states both on the even and on the odd indices. Let k be the first index for which some earlier index $j < k$ of the *same parity* points to the same state: $r_j = r_k$. If k is odd, j is also odd and so $j \geq 1$; then $r_j = r_k \implies \rho^{-1}(r_j) = \rho^{-1}(r_k) \implies r_{j-1} = r_{k-1}$ and $k-1$ also has the property that k is the earliest one to have, a contradiction. So, k must be even, and so must j. In fact, j must be 0 —or we reach a contradiction, as before, with π^{-1} instead of ρ^{-1}. Hence, the first state to be revisited is $r_0 = q$ and the trip consists of infinitely many copies of a list $r_0, r_1, \ldots, r_{k-1}$, for some even $k \geq 2$ and with no two states being both equal and at indices of the same parity.

Overall, either we reach a state $r_k \in R$ that is ρ-mapped to a state $q' \in R'$ or we return to the starting state $q \in R$ having repeated no state in L and no state

[7] Note that we abuse notation here. Bijection ρ can only return a *pair* of the form $(p, \mathtt{l})$ or $(p, \mathtt{r})$. So, in the description (1) above, $\rho(\cdot)$ really means 'the first component of $\rho(\cdot)$, if the second component is $\mathtt{l}$'. Similarly, 'ρ fails to return a state in L' means 'ρ returns a pair of the form $(p, \mathtt{r})$'. Hopefully, the abuse does not confuse.

in R. We define $\sigma : R \to (R')_\perp$ to encode exactly this information: in Case 1, $\sigma(q) = q'$; in Case 2, $\sigma(q) = \perp$. In either case, our trip respects π and ρ, which in turn respect the behavior of N on $\vdash w_1 w_2 \dots w_{i-1} a$. So, clearly: $\sigma(q) = q'$ implies some $c \in \text{RCOMP}_{N,q}(\vdash w_1 \cdots w_{i-1} a)$ respects π, ρ and *hits right into* q'; $\sigma(q) = \perp$ implies some *looping* $c \in \text{RCOMP}_{N,q}(\vdash w_1 \cdots w_{i-1} a)$ respects π, ρ and *visits only states from R when under a*. This concludes the definition of σ. $\square$

We can now define π'. We examine three cases about its argument.

(a) some $p \in L'$ that is ρ-mapped to an $r \in R'$. Then we just let $\pi'(p) = r$.

(b) some $p \in L'$ that is ρ-mapped to an $r \in L$. Then we consider $q = \pi(r)$. We know N can start at p under a and eventually reach q under a, so we ask what can happen after that if we keep following ρ and π. We examine $\sigma(q)$.

If $\sigma(q) = \perp$, then we will return to q after a cycle of length ≥ 2, having visited only states of R when under a. But can this happen? If it does, then the next-to-last and last steps in this cycle will follow ρ and π respectively, ending in q. Since ρ, π are bijections, the last two states (before q) in this cycle must respectively be p and r. In particular, p must appear in the cycle under a. But, since the cycle stays within R whenever under a, we must have $p \in R$, and hence R and L' intersect. But then (L, R), (L', R') are not compatible, a contradiction.

Hence, we know $\sigma(q) = q' \in R'$. And we can safely set $\pi'(p) = q'$.

(c) the special value $\perp$. The reasoning is similar to the previous case. We consider $q = \pi(\perp)$ and examine $\sigma(q)$. Again, $\sigma(q) = \perp$ is impossible, as it would imply $\perp \in L$. Hence, $\sigma(q) = q'$ for some $q' \in R'$ and we can safely set $\pi'(\perp) = q'$.

This concludes the definition of π'. It is easy to check π' satisfies the conditions of the claim. Hence, the inductive step is complete, as is the overall proof.

The Construction. We now describe the 1NFA N' simulating N. By Lemma 2 and the main observation, N' need simply check if some sequence of frontiers of N fits the input. So, N' just 'guesses' such a sequence. This needs 1 state per frontier, and a standard argument shows N has exactly $\binom{2n}{n+1}$ of them.

4 The Lower Bound

In this section, we exhibit an n-state 2NFA N that has no equivalent 1NFA with fewer than $\binom{2n}{n+1}$ states. In fact, N will even be *deterministic* and *single-pass*.

The Witness. Fix $n \geq 1$ and consider the alphabet $\Gamma = ([n] + ([n] \to [n])) \times \{\texttt{l}, \texttt{r}\}$. Of all strings in Γ^*, we will only care about the ones following the pattern

$$(x, \texttt{l})(g, \texttt{l})(h, \texttt{r})(y, \texttt{r}) \tag{2}$$

where $x, y \in [n]$, g and h are partial functions from $[n]$ to $[n]$, and $h(y)$ is undefined. We call these strings *nice inputs*. Intuitively, given a nice input as (2), we think of the graph of Fig. 5b, where the columns are two copies of $[n]$, the arrows between them are determined by g (left-to-right) and h (right-to-left), and the two special nodes by x (entry point) and y (exit). In this graph, a path from x to y may or may not exist; if it does, we say the nice input 'has a path'.

Letting $\mathit{\Pi}_{\text{yes}}$ (respectively, $\mathit{\Pi}_{\text{no}}$) be the set of nice inputs that (do not) have a path, we can easily see that the promise problem[8] $\mathit{\Pi} = (\mathit{\Pi}_{\text{yes}}, \mathit{\Pi}_{\text{no}})$ can be solved by a single-pass 2DFA with state set $[n]$. This is our witness, N.

Intuition. Consider an arbitrary frontier $F = (L, R)$ of N and list the elements of $L, R \subseteq [n]$ in increasing order, $L = \{x_1, \ldots, x_m\}$ and $R = \{y_1, \ldots, y_{m+1}\}$, for the appropriate $0 \le m < n$. Since $m < n$, we know $L \ne [n]$ and we can name an element outside L, say $x_0 = \min \overline{L}$. Then the combined list

$$x_0\, y_1\, x_1\, y_2\, x_2\, \cdots\, y_m\, x_m\, y_{m+1} \tag{3}$$

gives rise to the nice input $w_F = (x_F, \mathtt{l})(g_F, \mathtt{l})(h_F, \mathtt{r})(y_F, \mathtt{r})$ (see Fig. 6a), where $x_F = x_0$, function g_F maps every x in the list (3) to the following y, function h_F maps every $y \ne y_{m+1}$ to the following x, and $y_F = y_{m+1}$:

$$\begin{aligned} x_F &= \min \overline{L}, & y_F &= \max R, \\ g_F &= \{(x_i, y_{i+1}) \mid 0 \le i \le m\}, & h_F &= \{(y_i, x_i) \mid 1 \le i \le m\}. \end{aligned} \tag{4}$$

It is easy to verify that, *for any frontier F of N, the computation of N on w_F is accepting and its frontier under the middle boundary is exactly F.* This implies that, if N' is the 1NFA constructed for N as in Sect. 3, then *every* state of N' is used in *some* accepting computation. Which suggests N' is minimal.

The Proof. Every two frontiers F, F' of N give rise to the nice input (Fig. 6c)

$$w_{F,F'} = (x_F, \mathtt{l})(g_F, \mathtt{l})(h_{F'}, \mathtt{r})(y_{F'}, \mathtt{r}),$$

where x_F, g_F, $h_{F'}$, $y_{F'}$ are defined by (4). Crucially, in the $\binom{2n}{n+1} \times \binom{2n}{n+1}$ matrix $W = [w_{F,F'}]_{F,F'}$ containing all such inputs, two distinct strings at cells symmetric with respect to the main diagonal cannot both have a path.

Claim. For F, F' two frontiers of N: $w_{F,F'}, w_{F',F} \in \mathit{\Pi}_{\text{yes}} \iff F = F'$.

Proof. [$\Leftarrow$] Trivial. [$\Rightarrow$] Suppose $F = (L, R)$ and $F' = (L', R')$. We assume that $F \ne F'$ and prove that at least one of $w_{F,F'}$, $w_{F',F}$ lacks a path.

Let $m = |L|$, $m' = |L'|$ and consider the lists defined by F and F', as in (3):

$$x_0\, y_1\, x_1\, y_2\, x_2\, \cdots\, y_m\, x_m\, y_{m+1} \quad \text{and} \quad x'_0\, y'_1\, x'_1\, y'_2\, x'_2\, \cdots\, y'_{m'}\, x'_{m'}\, y'_{m'+1}.$$

If these were identical *after* their first elements, they would agree in their lengths, their x's (except possibly x_0, x'_0), and their y's, forcing $F = F'$, a contradiction. So, there must be positions of disagreement after 0. Consider the earliest one.

If this position is occupied by y's, say y_i and y'_i, then *either* $y_i < y'_i$ (Case 1) *or* $y_i > y'_i$ (Case 2). If it is occupied by x's, say x_i and x'_i, then *either* $x_i < x'_i$ or x'_i is not present at all[9] (Case 3) *or* $x_i > x'_i$ or x_i is not present at all (Case 4).

[8] By a (*promise*) *problem* over $\mathit{\Sigma}$ we mean a pair $\mathit{\Pi} = (\mathit{\Pi}_{\text{yes}}, \mathit{\Pi}_{\text{no}})$ of disjoint subsets of $\mathit{\Sigma}^*$. An automaton *solves* $\mathit{\Pi}$ iff it accepts *every* $w \in \mathit{\Pi}_{\text{yes}}$ but *no* $w \in \mathit{\Pi}_{\text{no}}$, while arbitrary behavior is allowed on strings outside $\mathit{\Pi}_{\text{yes}} + \mathit{\Pi}_{\text{no}}$.

[9] This happens if the list for F' stops at y'_i.

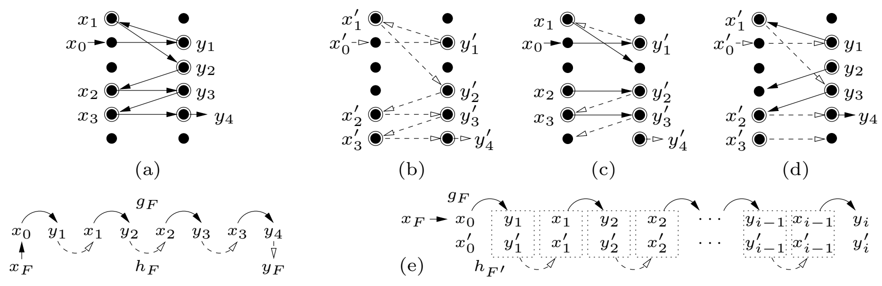


Fig. 6. (a) The input w_F when $n = 6$ and $F = (\{1, 4, 5\}, \{2, 3, 4, 5\})$, and how to derive it from the list $2, 2, 1, 3, 4, 4, 5, 5$. (b) A new input $w_{F'}$, for $F' = (\{1, 5, 6\}, \{2, 4, 5, 6\})$. (c,d) Inputs $w_{F,F'}$ and $w_{F',F}$. (e) Proving that at most one of them has a path.

We present the argument for Case 1 —the rest are similar. So, suppose the first disagreement is $y_i < y'_i$. Then all previous positions after 0 contain identical elements (Fig. 6e). Also, y_i is not in R': if it were, then it would be in the sublist $y'_1, \ldots, y'_{i-1}$ (since $y_i < y'_i$), and hence in $y_1, \ldots, y_{i-1}$ (the two sublists coincide), a contradiction (since $y_1, \ldots, y_{i-1} < y_i$). So $y_i \notin R'$. Therefore $y_i \neq y_{F'}$ and $h_{F'}(y_i)$ is undefined. But then, searching for a path in $w_{F,F'}$, we travel

$$x_0 \xrightarrow{g_F} (y_1 = y'_1) \xrightarrow{h_{F'}} (x'_1 = x_1) \xrightarrow{g_F} (y_2 = y'_2) \xrightarrow{h_{F'}} \cdots \xrightarrow{h_{F'}} (x'_{i-1} = x_{i-1}) \xrightarrow{g_F} y_i$$

reaching a node which is neither the exit $y_{F'}$ nor the start of an $h_{F'}$-arrow. $\square$

Now suppose a 1NFA A solves Π with fewer than $\binom{2n}{n+1}$ states. For each frontier F of N, we know $w_F = w_{F,F}$ is in Π_{yes}, so A accepts it. Pick an accepting $c_F \in \text{COMP}_A(w_F)$ and let q_F be the state right after the middle boundary is crossed. By the small size of A, we know $q_F = q_{F'}$ for some $F \neq F'$. But then, a standard cut-and-paste argument on c_F, $c_{F'}$ shows A also accepts $w_{F,F'}$ and $w_{F',F}$. Since both are nice inputs, we have $w_{F,F'}, w_{F',F} \in \Pi_{\text{yes}}$, contradicting the last claim.

5 Conclusion

We have shown the exact trade-off in the conversion from 2NFAs to 1NFAs. Our argument complemented that of Birget [12] by carefully removing some redundancies in its constructions. Crucially, the simulation performed by our optimal 1NFA is as 'meaningful' as the simulation given in [2] for the removal of nondeterminism from 1NFAs: each state corresponds to a (realizable and irreplaceable, as an instantaneous description) set-theoretic object that naturally 'lives' in the computations of the simulated 2NFA. Frontiers also allowed a set-theoretic *characterization of* 2NFA *acceptance* (already present in [12], essentially) that complements the set-theoretic *characterization of* 2NFA *rejection* given by Vardi [19]. Finally, by applying the concept of promise problems even to regular languages, we nicely confirmed its reputation for always leading us straight to the combinatorial core of the hardness of a computational task.

We do not know if the large alphabet size of problem Π is neccessary for the exactness of this trade-off. Also, it would be interesting to have the exact trade-offs in the conversions towards and from other types of automata (e.g., alternating, probabilistic) or more powerful machines (e.g., pushdown automata).

Many thanks to J.C. Birget for his help with some references; also, for raising our understanding of the subject to a level from which exact solutions could be seen.

References

1. Rabin, M.O., Scott, D.: Remarks on finite automata. In: Proceedings of the Summer Institute of Symbolic Logic, Cornell (1957) 106–112
2. Rabin, M.O., Scott, D.: Finite automata and their decision problems. IBM Journal of Research and Development **3** (1959) 114–125
3. Ott, G.: On multipath automata I. Research report 69, SRRC (1964)
4. Meyer, A.R., Fischer, M.J.: Economy of description by automata, grammars, and formal systems. In: Proceedings of the Symposium on Switching and Automata Theory. (1971) 188–191
5. Moore, F.R.: On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic, and two-way finite automata. IEEE Transactions on Computers **20** (1971) 1211–1214
6. Seiferas, J.I.: Manuscript communicated to M. Sipser (October 1973)
7. Sakoda, W.J., Sipser, M.: Nondeterminism and the size of two-way finite automata. In: Proceedings of the Symposium on the Theory of Computing. (1978) 275–286
8. Leung, H.: Separating exponentially ambiguous finite automata from polynomially ambiguous finite automata. SIAM Journal of Computing **27** (1998) 1073–1082
9. Rabin, M.O.: Two-way finite automata. In: Proceedings of the Summer Institute of Symbolic Logic, Cornell (1957) 366–369
10. Shepherdson, J.C.: The reduction of two-way automata to one-way automata. IBM Journal of Research and Development **3** (1959) 198–200
11. Hopcroft, J.E., Ullman, J.D.: Introduction to automata theory, languages, and computation. Addison-Wesley, Reading, MA (1979)
12. Birget, J.C.: State-complexity of finite-state devices, state compressibility and incompressibility. Mathematical Systems Theory **26** (1993) 237–269
13. Damanik, D.: Finite automata with restricted two-way motion. Master's thesis, J. W. Goethe-Universität Frankfurt (1996) (In German.)
14. Birget, J.C.: Two-way automata and length-preserving homomorphisms. Report 109, Department of Computer Science, University of Nebraska (1990)
15. Barnes, B.H.: A two-way automaton with fewer states than any equivalent one-way automaton. IEEE Transactions on Computers **C-20** (1971) 474–475
16. Sipser, M.: Lower bounds on the size of sweeping automata. Journal of Computer and System Sciences **21** (1980) 195–202
17. Kapoutsis, C.: Deterministic moles cannot solve liveness. (In preparation.)
18. Goldstine, J., Kappes, M., Kintala, C.M.R., Leung, H., Malcher, A., Wotschke, D.: Descriptional complexity of machines with limited resources. Journal of Universal Computer Science **8** (2002) 193–234
19. Vardi, M.Y.: A note on the reduction of two-way automata to one-way automata. Information Processing Letters **30** (1989) 261–264